



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/660,534	09/12/2003	Martin Soukup	71493-1141 /aba	5333
7380	7590	02/26/2007	EXAMINER	
SMART & BIGGAR P.O. BOX 2999, STATION D 900-55 METCALFE STREET OTTAWA, ON K1P5Y6 CANADA			INGBERG, TODD D	
			ART UNIT	PAPER NUMBER
			2193	
SHORTENED STATUTORY PERIOD OF RESPONSE		MAIL DATE	DELIVERY MODE	
3 MONTHS		02/26/2007	PAPER	

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No. 10/660,534	Applicant(s) SOUKUP, MARTIN	
	Examiner Todd Ingberg	Art Unit 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 30 November 2006.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-15 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-25 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 12 September 2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date <u>12/1/06</u> . | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

Claims 1 – 15 have been examined.

Claims 1, 5, 6, 11, 12, 13, 14 and 15 have been amended.

Information Disclosure Statement

1. Information Disclosure Statements were December 1, 2006 has been considered.

Specification

2. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed. Legal words like “method” and “apparatus” as present should be removed. Applicant has argued the use is common. Examiner does not find it descriptive and invites Applicant to participate in naming the title. If the case reaches allowance the Examiner will name the invention.

3. The amendment to Specification using dashes for instantiation has been entered.

Claim Rejections - 35 USC § 101

4. Prior rejection under 35 U.S.C. 101 has been overcome by amendment.

Claim Objections

5. Prior claims objections have been overcome by amendment.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Art Unit: 2193

7. Claims 1- 6 and 10 -15 are rejected under 35 U.S.C. 102(b) as being anticipated by Template Software.

The **Template** product line contains:

The SNAP programming language

The Workflow Template

The Web Component

These three layered products work together.

The documentation sets for the products contains the following manuals.

SNAP released June 1997

SNAP Language Reference (Not used in this Office Action)

Using the SNAP Language (Not used in this Office Action)

Using the SNAP Communication Component (Referred to as **COM**)

Using the SNAP Graphic User Interface Component (Not used in this Office Action)

Getting Started with SNAP (Not used in this Office Action)

Using the SNAP Display Editors (Not used in this Office Action)

SNAP Class Library Reference (Not used in this Office Action)

Using the SNAP External Application Software Component (Not used in this Office Action)

Using the SNAP Development Environment (Referred to as **SNAP**)

SNAP Module Library Reference (Not used in this Office Action)

Using the SNAP Permanent Storage Component (Referred to as **PREM**)

Workflow released September 1997

Developing a WFT Workflow System (Not used in this Office Action)

Art Unit: 2193

Using the WFT Development Environment (Not used in this Office Action)

WFT Library Reference (Not used in this Office Action)

Web Component

Using the Web Component (Referred to as **WEB**)

SNAP Training Manual – (Referred to as **Train**)

Since, these products work together they constitute a single reference and can be used as the basis for a rejection based on anticipated by a product offering.

Claim 11

Template anticipates a system adapted to assist with the migration of a software application from a first interface to a second interface (SNAP, Chapters 5 – 7, PERM, Chapters 2 – 4, COM, Chapters 3 – 7 – teach the SIB connection and Schema Mapping – see COM, page 4-2 to 4-3 for overview of class model and PERM, page 3-11 for basics on Mapping and for details on the OO tools involved see SNAP, pages 5-8 to 5-10 for SIB editor and SNAP, page 6-10 for Schema mapping), the system comprising a processing platform and a computer-readable medium comprising auto-generation software (Changing the SIB connection as per above), the system being adapted to receive a computer-readable mapping from the first interface to the second interface (PERM, pages 3-28 to 3-29 – Changing the SIB connection and leaving the mapping in place – e.g. change from ODBC SIB connection to Oracle), the processing platform being adapted to execute instructions of the auto-generation software to process the mapping to output an interface wrapper wherein the interface wrapper allows the software application to transparently communicate with the second interface (New SIB connection and same mapping as per above – also see as brought out in arguments by Applicant –SNAP, page 5-9 last two Commands).

Claim 12

The system of claim 11, wherein the instructions of the auto-generation software (Template is object oriented including SIB of claim 11) comprises the following instructions:

a) each class in the first interface (SNAP, Chapter 6, Database Mapping Editor – see page 6-10 also see as brought out in arguments by Applicant –SNAP, page 5-9 last two Commands)

a-1) from all classes to be included in the interface wrapper mapped front the class in the first interface, select a master class to hold handles to all other classes in the interface wrapper mapped from the class in the first interface (SNAP, page 6-10);

b) for each class to be included in the interface wrapper as set out in the computer-readable mapping (SIB and Schema mapping of claim 11)

b-1) if the class to be included in the interface wrapper is the master class, initialize all the handles in the master class; (SIB connection is object oriented – instantiation by definition part of OO)

Art Unit: 2193

b-2) for each attribute in the class to be included in the interface wrapper (SNAP, page 6-10, attr in Figure)

b-2-1) add code from the computer-readable mapping related to the attribute to the class to be included in the interface wrapper (SNAP, page 6-10, Mapping in Figure);

b-3 for each method in the class to be included in the interface wrapper (SNAP, page 6-10, part of class Mapped by definition classes are attributes and methods)

b-3-1) add code from the computer-readable mapping related to the method to the class to be included in the interface wrapper (SNAP, page 6-10 to 6-11).

Claim 13

The system of claim 11 wherein the instructions from the auto generation software comprises the following instructions: -for each class to be included in the interface wrapper as set out in the computer-readable mapping:

a) define the class to be included in the interface wrapper

b) for each class from the first interface mapped to the class to be included in the interface wrapper as set out in the computer readable mapping:

b-1) if-upon determining that the class from the first interface is mapped only to the class to be included in the interface wrapper:

b-1-1) add a member for the class from the first interface to the class to be included in the interface wrapper;

b-1-2) add construction of the member to all constructor in the class to be included in the interface wrapper;

b-2) if-upon determining, that the class from the first interface is not mapped only to the class to be included in the interface wrapper an upon determining that the class to be included in the interlace wrapper is the first of all classes in the interface wrapper mapped from the class from the first interface as set out in the computer-readable mapping:

b-2-1') designate the class to be included in the interface wrapper as a master class to hold handles to all other classes in the interface wrapper mapped from the class in the first interface;

b-2-2) add a member for the class from the first interface to the class to be included in the interface wrapper;

b-2-3) add construction of the member to all constructors in the class to be included in the interface wrapper;

b- 2-4) for each other class in the interface wrapper mapped from the class from the first interface

b-2-4-1) add a member for the other class in the interface wrapper to the master class;

b-2-4-2) add construction of the other class in the interface wrapper to all constructors in the master class;

b-2-4-3) add a call to initialize the member wherein the member will know that the class to be included in the interface wrapper is the master class in all constructors of the other class in the interface wrapper;

b-2-4-4) add a method to the other class in the interface wrapper to retrieve the member by a type name of the other class;

b-3) eon determining that the class from the first interface is not mapped only to the class to be included in the interface wrapper and upon determining that the class to be included in the

Art Unit: 2193

interface wrapper is not the first of all classes in the interface wrapper mapped from the class from the first interface as set out in the computer readable mapping:

b-3-1) add a member for the master class to the class to be included in the interface wrapper;

b-3-2) add a method to the class to be included in the interface wrapper to allow the member to be initialized to point to the master class;

b-3-3) add a method to the class to be included in the interface wrapper to retrieve the member,

c) for each attribute in the class to be included in the interface wrapper

c-1) add code from the computer-readable mapping related to the attribute to the class to be included in the interface wrapper;

d) for each method in the class to be included in the interface wrapper

d-1) add code from the computer-readable mapping related to the method to the class to be included in the interface wrapper. As per the rejections for claims 11 and 12.

Claim 14

A computer readable medium containing instructions for automatically generating and outputting an interface wrapper to facilitate migration of a software application from a first interface to a second interface, wherein, given a computer-readable mapping from the first interface to the second interface, the instructions comprise: a) for each class in the first interface

a) from all classes to be included in the interface wrapper mapped from the class in the first interface, select a master class to hold handles to all other classes in the interface -wrapper mapped from the class in the first interface;

b) for each class to be included in the interface wrapper as set out in the computer-readable mapping

b-1) if the class to be included in the interface wrapper is the master class, initialize all the handles in the master class;

b-2) for each attribute in the class to be included in the interface wrapper

b-2-1) add code from the computer-readable mapping related to the attribute to the class to be included in the interface wrapper; for each method in the class to be included in the interface wrapper

b 3-1) add code from the computer-readable mapping related to the method to the class to be included in the interface wrapper. As per the rejections for claims 11 and 12.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Art Unit: 2193

9. Claims 1 – 6, 10 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Template Software in view of “Object-Oriented Information Systems, Planning and Implementation”, by David A. Taylor, published April 10, 1992.

Claim 1

Template teaches a software application that uses a first interface related to a first piece of software code (SNAP, page 2- 4 creating applications that communicate also see SNAP, page 3-44, Parameters for signature for the Functions), a method of migrating the software application to allow the software application to use a second interface instead of the first interface (Template is object oriented – principle of polymorphism is part of OO by definition – **Train**, page 1-16), the method comprising:

creating a computer-readable mapping between the first interface and the second interface SNAP, page 2- 4 creating applications that communicate also see SNAP, page 3-44, Parameters for signature for the Functions);

running the mapping through an auto-generator, wherein the auto-generator uses the mapping to automatically generate an interface wrapper (SNAP, 2-41, auto-parse for CD files);

outputting the interface wrapper to replace the first interface and the first piece of software code (result of steps above), thereby interposing the interface wrapper between the software application and the second interface (Interpreted to be ability to “call”); wherein the interface wrapper allows the software application to communicate with the second interface instead of the first interface (Polymorphism above – where the new function is used as determined by the parameters passed (overriding) and the other is not called).

Template provides the development environment for building object-oriented software. Template does not explicitly use the term wrapper. It is Taylor who explicitly teaches the word and application of the common use of a wrapper in object-oriented software development (Taylor, page 296). Therefore, it would have been obvious to one of ordinary skill to utilize the object oriented development environment to implement a wrapper as Taylor teaches. Because, wrappers, “.. offer a relatively painless path to integration.”(Taylor, page 297).

Claim 2

The method of claim 1, wherein the interface wrapper provides for at least one of
(a) forward compatibility wherein the interface wrapper allows the software application to transparently communicate with the second interface and
(b) backward compatibility wherein the interface wrapper allows the second interface to transparently communicate with the software application. (Taylor, teaches a “wrapper” see pages 296 – 297, focus on page 297 – third paragraph last sentence)

Claim 3

The method of claim 2, wherein for forward compatibility, creating the computer-readable mapping comprises creating a mapping from the first interface to the second interface. (Taylor, “wrapper” see page 297 – third paragraph last sentence – back and forth).

Claim 4

The method of claim 2, wherein for backward compatibility, creating the computer-readable mapping comprises creating a mapping from the second interface to the first interface. As per claim 3.

Claim 5

The method of claim 1, wherein the auto-generator, in an object oriented environment, is software (as per claim 1) comprising the following instructions:

a) for each class in the first interface

1) from all classes to be included in the interface wrapper mapped from the class in the first interface, select a master class to hold handles to all other classes in the interface wrapper mapped from the class in the first interface (As per claim 1 – see the object model editor SNAP, pages 3-6 to 3-9);

b) for each class to be included in the interface wrapper as set out in the computer-readable mapping (In the citing above – Inheritance is part of OO by definition – the ability to link classes and have all methods and attributes linked through an inheritance model, SNAP, page 3-11).

1) if the class to be included in the interface wrapper is the master class (depending on author also known as parent class or base class), initialize all the handles in the master class (Instantiation – is part of OO by definition – where the handlers are initialized in the classes – see SNAP, page A-3) ;

2) for each attribute in the class to be included in the interface wrapper (Inheritance as mentioned above includes attributes and methods)

2-1) add code from the computer-readable mapping related to the attribute to the class to be included in the interface wrapper;

3) for each method in the class to be included in the interface wrapper (As per above)

3-1) add code from the computer-tradable mapping related to the method to the class to be included in the interface wrapper (SNAP, Inheritance, page 3-70).

Claim 6

The method of claim 1, wherein the auto-generator, in an object-oriented environment, is software comprising the following instructions

for each class to be included in the interface wrapper as set out in the computer-readable mapping:

a) define the class to be included in the interface wrapper

b) for each class from the fast interface mapped to the class to be included in the interface wrapper as set out in the computer readable mapping:

b-1) upon determining that the class from the first interface. is mapped only to the class to be included in the interface wrapper

b-1-1) add a member for the class from the first interface to the class to be included in the interface wrapper;

b-1-2) add construction of the member to all constructors in the class to be included in the interface wrapper;

Art Unit: 2193

b2) upon determining that the class from the first interface is not mapped only to the class to be included in the interface wrapper and upon determining that the class to be included in the interface wrapper is the first of all classes in the interface wrapper mapped from the class from the first interface as set out in the computer readable mapping:

b2-1) designate the class to be included in the interface wrapper as a master class to hold handles to all other classes in the interface wrapper mapped from the class in the first interface;

b2-2) add a member for the class from the first interface to the class to be included in the interface wrapper,

As per claim 5.

b2-3) add construction of the member to all constructors in the class to be included in the interface wrapper (Constructors are inherent in OO the ability to instantiate new objects – often the command “new” is part of the language),

b2-4) for each other class in the interface wrapper mapped from the class from the first interface

b2-4-1) add a member for the other class in the interface wrapper to the master class;

b2-4-2) add construction of the other class in the interface wrapper to all constructors in the master class;

b2-4-3) add a call to initialize the member wherein the member will know that the class to be included in the interface wrapper is the master class in all constructors of the other class in the interface wrapper;

As per claim 5 above - inheritance

b2-4-4) add a method to the other class in the interface wrapper to retrieve the member by a type name of the other class (SNAP, page 11-18, Getters – part of accessor functions);

b-3) upon determining that the class from the first interface is not mapped only to the class to be included in the interface wrapper and upon determining that the class to be included in the interface wrapper is not the first of all classes in the interface wrapper mapped from the class from the first interface as set out in the computer readable mapping (SNAP, page 3-22, multiple inheritance):

b-3-1) add a member for the master class to the class to be included in the interface wrapper, (inheritance as per above)

b-3-2) add a method to the class to be included in the interface wrapper to allow the member to be initialized to point to the master class (initialize classes as per above);

b-3-3) add a method to the class to be included in the interface wrapper to retrieve the member;

c) for each attribute in the class to be included in the interface wrapper (inheritance)

c-1) add code from the computer-readable mapping related to the attribute to the class to be included in the interface wrapper;

d) for each method in the class to be included in the interface wrapper

d-1) add code from the computer-readable mapping related to the method to the class to be included in the interface wrapper.

As per claim 5 above - inheritance

Claim 10

Art Unit: 2193

The method of claim 1, wherein the software application is migrated from the first interface to the second interface without modifying the software application. See claim 1 – polymorphism.

Claim 15

A computer-readable medium containing instructions for automatically generating and outputting an interface wrapper to facilitate migration of a software application from a first interface to a second interface, wherein, given a computer-readable mapping from the first interface to the second interface, the instructions comprise:

each class to be included in the interface wrapper as set out in the computer-readable mapping,

a) define the class to be included in the interface wrapper

b) for each class from the first interface mapped to the class to be included in the interface wrapper as set out in the computer readable mapping

b-1) A- upon determining that the class from the first interface is mapped only to the class to be included in the interface wrapper

b-1-1) add a member for the class from the first interface to the class to be included in the interface wrapper;

b-1-2) add construction of the member to all constructors in the class to be included in the interface wrapper

b-2) upon determining , that the class from the first interface is not mapped only to the class to be included in the interface wrapper and upon determining that the class to be included in the interface wrapper is the first of all classes in the interface wrapper mapped from the class from the first interface as set out in the computer readable mapping

b-2-1) designate the class to be included in the interface wrapper as a master class to hold handles to all other classes in the interface wrapper mapped from the class in the first interface;

b-2-2) add a member for the class from the first interface to the class to be included in the interface wrapper;

b-2-3) add construction of the member to all constructors in the class to be included in the interface wrapper;

b-2-4) for each other class in the interface wrapper mapped from the class from the first interface

b-2-4-1) add a member for the other class in the interface wrapper to the master class;

b-2-4-2) add construction of the other class in the interface wrapper to all constructors in the master class;

b-2-4-3) add a call to initialize the member wherein the member will know that the class to be included in the interface wrapper is the master class in all constructors of the other class in the interface wrapper;

b-2-4-4) add a method to the other class in the interface wrapper to retrieve the member by a type name of the other class;

b-3) determining that the class from the first interface is not mapped only to the class to be included in the interface wrapper and upon determining that the class to be included in the interface wrapper is not the first of all classes in the interface

wrapper mapped from the class from the first interface as set out in the computer readable mapping

b-3-1) add a member for the master class to the class to be included in the interface wrapper;

Art Unit: 2193

b-3-2) add a method to the class to be included in the interface wrapper to allow the member to be initialized to point to the master class;

b-3-3) add a method to the class to be included in the interface wrapper to retrieve the member,

c) for each attribute in the class to be included in, the interface wrapper

c-1) add code from the computer-readable mapping related to the attribute to the class to be included in the interface wrapper;

d) for each method in the class to be included in the interface wrapper

d-1) add code from the computer-readable mapping related to the method to the class to be included in the interface wrapper. As per claims 1 and 5 above.

Template provides the development environment for building object-oriented software. Template does not explicitly use the term wrapper. It is Taylor who explicitly teaches the word and application of the common use of a wrapper in object-oriented software development (Taylor, page 296). Therefore, it would have been obvious to one of ordinary skill to utilize the object oriented development environment to implement a wrapper as Taylor teaches. Because, wrappers, "... offer a relatively painless path to integration." (Taylor, page 297).

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

11. Claims 7 – 9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Template and Taylor in view of Enterprise Application Integration with XML and JAVA, by J.P. Morgenthal et al, 2001, Chapter 5.

Claim 7

The method of claim 1, wherein the software application, the first interface and the second interface are written in the Java language. Template teaches the use of the object oriented programming language SNAP (SNAP, Chapter 5, 6 and 7) and Taylor explicitly teaches wrappers but they do not teach the use of the programming language JAVA.

It is XML that teaches the use of JAVA for application integration (XML, page 97). Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to alter Template to replace the SNAP language with JAVA, for implementation of wrappers as taught by Taylor, because non proprietary programming languages are more marketable.

Art Unit: 2193

Claim 8

The method of claim 1, wherein the computer readable mapping is written in a language selected from one of XML and UML. Template teaches the use of HTML (Web, page 4-33) not XML. And Taylor teaches a common use of wrappers. It is XML that teaches the use of XML for application integration (XML, page 97). Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to alter Template to replace the HTML language with XML to implement wrappers, because being able to define your own tags makes hypertext markup languages for flexible.

Claim 9

The method of claim 1, wherein the auto-generator is software written in a form selected from one of XSL templates, bean script and XDoclet. As per claim 8 – XSL templates are inherent in XML.

Response to Arguments

12. The following is a summary of Applicant's arguments beginning on page 18 of the response with the Examiner's response.

Applicant states "key features" are missing from the commercial product by Template Software.

- mapping between two interfaces
- auto-generator to read mapping automatically and replacing first interface and allow software application to use the second interface rather than the first.

Applicant identifies some of the components with in the Template references such as the SIB connector, Schema mapper and TCP/IP protocol and determines the two disparate process need this protocol to communicate.

Examiner's Response

The picture that the configuration of Template can not be on one machine is that of the applicant. If configured on more than one machine TCP/IP is one possible protocol that can be used. From this argument it appears Applicant's invention is limited to one machine. Template may be configured in many different environments including standalone. Applicant's picture of

Art Unit: 2193

needed TCP/IP is not accurate. The documentation covers the most common configuration (distributed).

Applicant Argument

Applicant has taken one figure page COM 5-5, Figure 5-4 and imposed a universal limitation on the product by stating only attributes can be shared.

Examiner's Response

Attributes to one of ordinary skill in the art is one a portion of an object by definition the methods are another portion. Examiner notes the Applicant has not addressed the methods that perform filter functions or calls (messaging gets or setters). In acknowledging the presence of methods the Applicant's argument is moot.

Applicant's Argument

SIB connection editor can not share classes.

Examiner's Response

When looking only at the connection itself one might not see class sharing. However, SNAP page 5-8 "Accessing the SIB Connection Editor" under class/object the mapping of classes is performed. Note that the mapping of data and methods (getters and setters, filter functions). When you change your import or export map you are changing the classes you share (see SNAP, page 5-9 last two Commands).

Applicant's Argument

Page 20 Applicant's depiction of how the schema mapper is reasonable accurate until they cite COM page 4-26 as a limit on the all, setting or retrieving, "... by a schema (i.e. classes, static attributes and methods can not be transferred by a schema."

Examiner's Response

Art Unit: 2193

Note the section Applicant cites is in the batch mode operations section. Not the runtime. Applicant has taken a limitation on batch processing and imposed it on the entire product.

Applicant's Argument

Bottom of page 20 - 21, Applicant is stating "Applicant states no ability to migrate between two interfaces is present and no "autogenerator" is not taught:

Examiner's Response

In view of the Examiner's responses above with the Applicant imposed limitation of only sharing data (not accurate). Changing out the import or export map changes more than data (filter functions, getter and setters in non batch mode). And accompanying interfaces. Template is also an object oriented environment. Changes to the objects such as import or export maps are considered autogenerated.

Applicant's Argument

Applicant argues the mapping is not taught between two interfaces.

Examiner's Response

Examiner contends with the details of the above responses the mapping should be quite evident.

Applicant's Argument

Other claims rely on the arguments of claim 1.

Examiner's Response

Non persuasive arguments for claim 1.

Summary

Applicant's statements about the commercial product Template Software are solely those of Applicant and do not reflect the position of the Patent Office.

Although, the claims are long when interpreted in the broadest reasonable interpretation in view of the Specification they read on the wrapping of a database with schema mapping (data and methods) with connection by a SIB connector (encompassing SIB object). In reviewing the rejection Applicant should revisit the methods (getters, setters, filter functions) and place themselves in the programmer role of updating with the purpose of changing the import/export map. Then place themselves in the SNAP environment as described below.

And Applicant should review the object model editor with class building in the normal use of building wrappers. Thoughts on the intended use of polymorphism and inheritance should be considered when amending. If Applicant can amend away from reading on the current rejection this is the area of the reference and object oriented programming the Applicant will want to consider (claims 1-6, 10, 15).

Conclusion

13. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

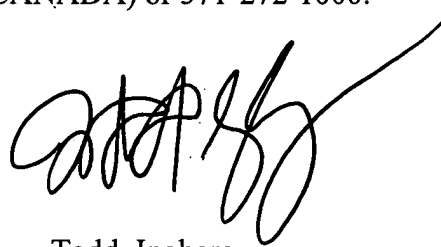
Correspondence Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Todd Ingberg whose telephone number is (571) 272-3723. The examiner can normally be reached on during the work week..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Art Unit: 2193

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

A handwritten signature in black ink, appearing to read 'TI', with a long, sweeping horizontal line extending to the right.

Todd Ingberg
Primary Examiner
Art Unit 2193

TI